How to Easily Print to PDF Using VBA Code

Authored by

stats writer

November 19, 2025

Generating a PDF document directly from an Excel workbook using VBA is a powerful automation technique that streamlines reporting and documentation processes. Instead of manually navigating the print dialogue, you can execute a simple script to handle the conversion efficiently. This method relies primarily on the **PrintOut** method, or more robustly, the ExportAsFixedFormat method, which is available on various objects, including the ActiveSheet object, allowing users to effortlessly export data while maintaining full control over output parameters.

This comprehensive guide will detail how to implement this functionality, providing clean, valid code examples that ensure your active sheet is converted into a high-quality, specified PDF file. We will explore the necessary arguments, such as defining the output file path, controlling the quality of the output, and managing how the document is handled post-publication. Mastering this technique is essential for anyone seeking to maximize efficiency and standardization within the Microsoft Office suite.

Understanding the VBA Print-to-PDF Mechanism

The core mechanism for converting Excel data into a static format like PDF resides in the **ExportAsFixedFormat** method. This method supersedes older, less flexible methods of file conversion and is the standardized approach for generating publication-ready documents. By calling this method on the appropriate Excel object (such as a `Worksheet`, `Range`, or `Workbook`), you instruct VBA to render the selected content into the specified fixed format.

When used with the ActiveSheet object, the command targets the worksheet currently visible and selected by the user, converting its entire printable area, or a user-defined range, into the output file. This provides immediate utility, allowing end-users to generate reports quickly based on their current view without needing to modify the underlying code.

The versatility of the **ExportAsFixedFormat** method also allows for granular control over the output. Parameters such as file quality, page range, and whether to include document properties can all be adjusted directly within the VBA code, ensuring the resulting PDF meets specific business or archival standards. This level of customization is what makes VBA automation superior to standard manual export procedures.

The Essential Syntax: ExportAsFixedFormat Method

To initiate the PDF printing process in VBA, we must define a clear subroutine and use the method with the mandatory and desired optional arguments. The following code structure demonstrates the basic implementation required to convert the currently active Excel sheet into a PDF file, saving it locally and opening it immediately.

You can use the following syntax in VBA to print the currently active Excel sheet to a PDF:

```
Sub PrintToPDF()

ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, _
Filename:="my_data.pdf", _
Quality:=xlQualityStandard, _
IncludeDocProperties:=False, _
IgnorePrintAreas:=False, _
OpenAfterPublish:=True

End Sub
```

This particular macro will print the currently active Excel sheet to a PDF called **my_data.pdf**. Since no specific file path is provided, it will be saved in the current folder where the active workbook resides.

Detailed Explanation of Key Parameters

The efficiency and customization of the PDF export process are controlled entirely by the arguments passed to the ExportAsFixedFormat method. Understanding these parameters is crucial for tailoring the output to specific needs, such as resolution, file size limitations, or post-export actions. Only the **Type** argument is mandatory for defining the output format.

The three most foundational parameters ensure proper output definition and naming:

Type (Required): This defines the type of file format for the export. For printing to PDF, this value must be set to the enumeration constant **xlTypePDF**. This is the only required argument to define the nature of the output file.
Filename (Optional but Recommended): This string argument specifies the name and, optionally, the full path where the resulting PDF file will be saved. **Note #1**: You can include a full file path in the **Filename** argument to save the PDF to a specific folder, overriding the default behavior of saving alongside the workbook.
Quality (Optional): This parameter dictates the rendering resolution. Using **xlQualityStandard** ensures high fidelity suitable for printing, whereas `xlQualityMinimum` can be used to generate smaller files optimized for web viewing.

Additional Boolean arguments control specific behaviors during the publishing process:

IncludeDocProperties (Optional): Controls whether metadata (author, title, keywords) is embedded in the PDF. Setting this to **False** often results in cleaner output for simple data exports.

IgnorePrintAreas (Optional): Setting this to **True** will force the export of the entire sheet content, ignoring any print ranges previously defined in Excel. Setting it to **False** (as shown in the example) respects the predefined print areas.
OpenAfterPublish (Optional): This argument provides excellent user feedback. **Note #2**: The line **OpenAfterPublish:= True** tells VBA to open the PDF using the system's default viewer as soon as it is exported. You can omit or set this argument to **False** if you don't want the PDF to be opened immediately after exporting.

Practical Implementation: Scope and Required Arguments

When working with the ExportAsFixedFormat method, it is crucial to understand the scope of the operation. Applying the method to the `ActiveSheet` ensures only the single, currently visible sheet is processed. If you need to export a specific range or multiple sheets, you would need to adjust the object reference accordingly (e.g., using `Sheets("Report").ExportAsFixedFormat` or iterating through a collection of sheets).

Furthermore, recognizing the minimal requirements of the method aids in writing efficient code. **Note #3**: The only required argument in the **ExportAsFixedFormat** method is **Type**, which must be set to **xlTypePDF** to print the sheet to a PDF format. While the other parameters are optional, defining the **Filename** is almost always necessary to track the output file effectively.

The ability to specify the file path dynamically is a powerful feature. By utilizing VBA functions like `ThisWorkbook.Path` combined with the desired filename, developers can create highly portable macros that always save the PDF in the same directory as the source workbook, regardless of where the user opens the file.

Example: Exporting an Excel Sheet to PDF

The following example shows how to utilize this macro in a real-world scenario. We will take a structured dataset and demonstrate the process of automated conversion, focusing on how formatting elements are preserved.

Suppose we have the following Excel sheet that contains information about various basketball players. This sheet is the active context when we run our routine:

	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**			
2	Mavs	22	12			
3	Heat	19	9			
4	Nets	14	4			
5	Rockets	20	6			
6	Rockets	30	5			
7	Mavs	34	7			
8	Mavs	30	7			
9	Heat	29	8			
10	Nets	14	6			
11	Nets	29	3			
12						
13						
14						
15						
16						
17						
18						
19						

Now suppose that we would like to export this specific sheet to a PDF called **my_data.pdf**, ensuring it is automatically opened for verification immediately after export.

Implementing the Export Macro

To perform the conversion, we use the previously defined subroutine. This code snippet ensures the ActiveSheet initiates the export with the necessary parameters defined:

We can create the following VBA macro to do so:

Sub PrintToPDF()

ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, _
Filename:="my_data.pdf", _
Quality:=xlQualityStandard, _
IncludeDocProperties:=False, _
IgnorePrintAreas:=False, _
OpenAfterPublish:=True

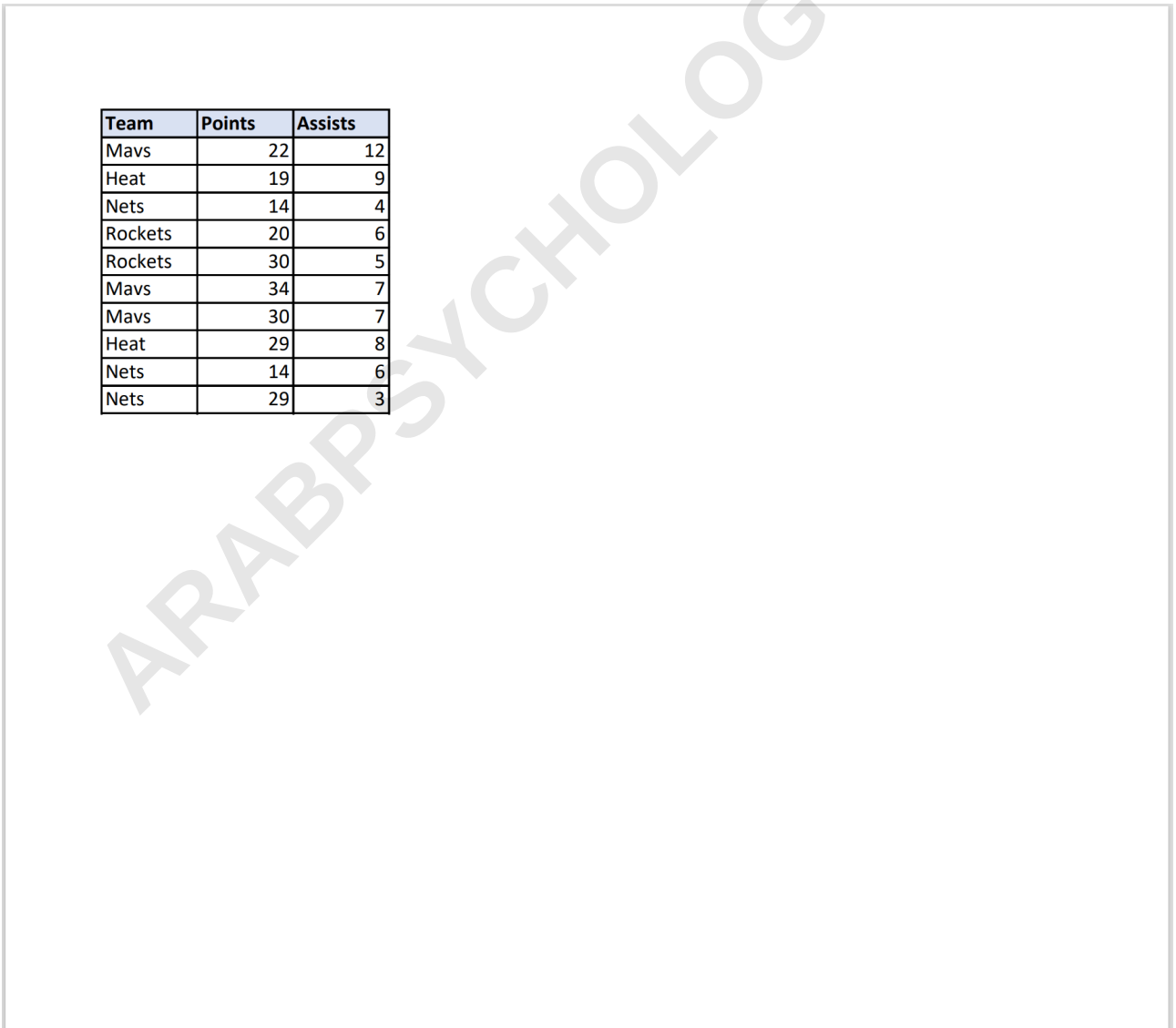
```
End Sub
```

When we run this macro, the Excel sheet is successfully exported to a PDF file, saved in the same folder as the workbook, and then the PDF is automatically opened for viewing, thanks to the `OpenAfterPublish:= True` parameter:

Examining the Code Output and Results

The system's execution of the ExportAsFixedFormat method results in a visually accurate Portable Document Format document. The primary success criterion is the preservation of formatting and layout from the source Excel sheet.

The resulting PDF document opened automatically will resemble this image:



Team	Points	Assists
Mavs	22	12
Heat	19	9
Nets	14	4
Rockets	20	6
Rockets	30	5
Mavs	34	7
Mavs	30	7
Heat	29	8
Nets	14	6
Nets	29	3

Note that the exact formatting of the cells with the borders and the fill color, which are crucial for readability, is included precisely in the PDF output. This demonstrates the high fidelity of the conversion process when utilizing the appropriate VBA method.

Troubleshooting and Further Documentation

If the macro fails, common causes include incorrect file paths (e.g., trying to save to a directory that does not exist), or permission issues preventing VBA from writing to the specified location. Always use robust error handling (like `On Error Resume Next`) in production code, although it is excluded here for simplicity.

For advanced scenarios, such as exporting only a specific range of cells instead of the entire sheet, you would apply the method to a `Range` object instead of the `ActiveSheet`, allowing precise control over the exported content. For example: `Range("A1:G10").ExportAsFixedFormat Type:=xlTypePDF, ...`

For comprehensive understanding and details on all possible optional parameters, including handling password protection or different page ranges (StartPage and EndPage arguments), it is recommended to review the official source material. **Note**: You can find the complete documentation for the **ExportAsFixedFormat** method in VBA on the Microsoft Learn platform, which serves as the authoritative guide for all object methods.